

MODELING PLATE AND SPRING REVERBERATION USING A DSP-INFORMED DEEP NEURAL NETWORK

Marco A. Martínez Ramírez, Emmanouil Benetos, Joshua D. Reiss

Centre for Digital Music, Queen Mary University of London, UK

ABSTRACT

Plate and *spring* reverberators are electromechanical systems first used and researched as means to substitute real room reverberation. Currently, they are often used in music production for aesthetic reasons due to their particular sonic characteristics. The modeling of these audio processors and their perceptual qualities is difficult since they use mechanical elements together with analog electronics resulting in an extremely complex response. Based on digital reverberators that use sparse FIR filters, we propose a signal processing-informed deep learning architecture for the modeling of artificial reverberators. We explore the capabilities of deep neural networks to learn such highly nonlinear electromechanical responses and we perform modeling of *plate* and *spring* reverberators. In order to measure the performance of the model, we conduct a perceptual evaluation experiment and we also analyze how the given task is accomplished and what the model is actually learning.

Index Terms— artificial reverberation, audio effects modeling, deep learning, sparse FIR.

1. INTRODUCTION

Reverberation occurs when delayed and attenuated copies of the direct sound appear as reflections. Each reflection is frequency dependent and defined by the directivity of the sound source and the physical attributes of the reflecting surfaces [1]. In the music and film industry, artificial reverberation was initially researched as a way of approximating the reflections occurring in room acoustics. This led to techniques that simulate reverberation, such as chamber, *plate*, *spring* and digital reverberators [2].

Plate reverberation is based on a large metal plate which vibrates due to a moving-coil transducer attached to its centre. This transducer is fed with an amplified dry input signal and the plate vibrations are read by a pickup sensor and further amplified [3]. The *plate* reverb sound is different from room acoustic reverberation and is characterized by a smooth noise-like response [4]. *Spring* reverberation is based on one or various helical springs suspended under low tension, attached to a magnetic bead and driven via an electromagnetic coupling [5]. The input audio source is transduced to spring vibrations which are read through a pickup sensor at the opposite end. The distinct sound of *spring* reverb is due to the various types of vibrations that occur, transverse and longitudinal, which cause a peculiar combination of wave and dispersive propagation [1].

Although originally developed as substitutes for room reverberators, digital implementations of these devices have been widely researched due to their distinctive sound which has been of great interest among musicians, music producers and sound engineers. *Plate*

reverberation has been emulated with different approaches such as numerical simulation techniques, where a finite difference scheme [6, 7, 8] or a modal description [9] is derived from the differential equations that describe the motion of the plate; and hybrid digital filter-based algorithms [10, 11, 12], where convolutional impulse responses and feedback delay networks are used to model the desired impulse response. Similarly, modeling of *spring* reverberation has been explored as wave digital filters [13], to explicitly model the wave and dispersive propagation; numerical simulation techniques such as finite difference schemes [14, 5, 15], and nonphysical modeling techniques [16, 17], where chains of allpass filters and varying delay lines are used to approximate the dispersive and reverberant features of *spring* reverb.

The modeling of these audio processors and their salient perceptual qualities remains an active research field. Their mechanical elements together with their analog circuitry yield a nonlinear and time-varying spatial system which is difficult to fully emulate digitally. Most of the methods are based on complete physical models or perceptual simplifications such as linearity and time-invariant behavior, thus, such models are not easily transferable to different artificial reverberators or cannot capture the full response of the system.

Deep learning architectures for black-box modeling of audio effects have been researched lately for linear effects such as equalization [18]; nonlinear memoryless effects such as tube amplifiers [19, 20, 21]; nonlinear effects with temporal dependencies such as compressors [22]; and linear and nonlinear time-varying effects such as flanging or ring modulation [23]. Deep learning for dereverberation has become a heavily researched field [24, 25], although applying artificial reverberation or modeling *plate* and *spring* reverb with deep neural networks (DNN) has not been explored yet.

Sparse FIR filtering has proven to be an efficient digital reverberation method [26, 2]. Dispersive reflections are approximated via FIR filters with sparsely placed coefficients, which are often determined by a pseudo-random number sequence such as velvet noise [27]. We incorporate these methods to model noise-like and dispersive responses, such as those present in *plate* and *spring* devices.

Prior to this work, end-to-end DNNs have not yet been implemented to model artificial reverberators, i.e. learning from input-output data and applying the reverberant effect directly to the dry input audio. In this paper, we use convolutional, recurrent and dense layers together with sparse FIR (SFIR) filters and time-varying mixing gains which coefficients are learnt by the network. We explore whether a deep learning architecture is able to emulate *plate* and *spring* reverberators and we measure the performance of the model through a listening test. Both perceptual and objective evaluations indicate that the proposed model successfully simulates the electromechanical devices and performs better than other DNNs for modeling audio effects.

2. METHODS

The Titan Xp GPU used for this research was donated by the NVIDIA Corporation. EB is supported by a RAEng Research Fellowship (RF/128). The *spring* reverb samples were recorded with the help of Giulio Moro.

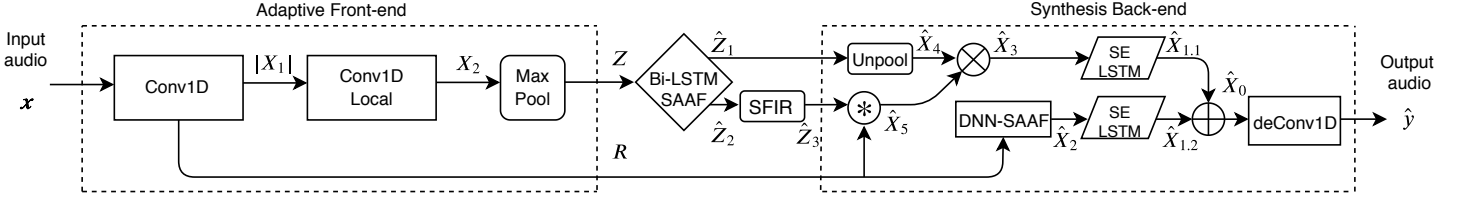


Fig. 1: Block diagram of the proposed model; adaptive front-end, latent-space and synthesis back-end.

2.1. Model

The model is completely based on time-domain input and works with raw and processed audio as input and output respectively. It is divided into three parts: adaptive front-end, latent-space and synthesis back-end. A block diagram is depicted in Fig. 1 and a more detailed structure can be seen online¹ together with the source code.

The **adaptive front-end** uses a filter bank architecture which learns the latent representation Z by performing time-domain convolutions with the input audio. It follows the same architecture as [23], where it contains two convolutional layers, one pooling layer and one residual connection. The front-end is considered adaptive since its convolutional layers learn a filter bank for each modeling task and directly from the audio. The model learns long-term memory dependencies by having an input x which consists of the current audio frame x concatenated with the ± 4 previous and subsequent frames. These frames are of size 4096 (256 ms) and sampled with a hop size of 50%.

The *Conv1D* layer has 32 one-dimensional filters of size 64 and is followed by the *absolute value* as nonlinear activation function. *Conv1D-Local* has 32 filters of size 128, each filter is locally connected and uses the *softplus* function as nonlinearity. This means we follow a filter bank architecture since each filter is only applied to its corresponding row in $|X_1|$: the output of *Conv1D*. The residual connection R is the corresponding row in X_1 and is the frequency band decomposition of the current input frame x . This is due the output of each filter of *Conv1D* can be seen as a frequency band. The *max-pooling* layer is a moving window of size 64, where the maximum value within each window corresponds to the output. All convolutional and pooling layers are time distributed, i.e. the same layer is applied to each of the 9 input frames from x .

The **latent-space** has as its main objective to process Z into two latent representations, \hat{Z}_1 and \hat{Z}_2 . The former corresponds to a set of envelope signals and the later is used to create the set of sparse FIR filters \hat{Z}_3 . It consists of two shared Bidirectional Long Short-Term Memory (Bi-LSTM) layers of 64, 32 units with the *hyperbolic tangent* as activation function and its output is fed to two independent Bi-LSTM layers of 16 units. Each of these layers is followed by a Smooth Adaptive Activation Function (SAAF) as the nonlinearity [28], obtaining in this way \hat{Z}_1 and \hat{Z}_2 . SAAFs consist of piecewise second order polynomials which can approximate any continuous function and are regularized under a Lipschitz constant to ensure smoothness. As shown in [19], SAAFs can be used as nonlinearities or waveshapers in audio processing tasks.

We propose a *SFIR* layer where we follow the constraints of sparse pseudo-random reverberation algorithms [2]. Nevertheless, instead of using discrete coefficient values such as -1 and $+1$, each coefficient can take any continuous value within that range. Each one of the coefficients is placed at a specific index position within each interval of T_s samples while all the other samples are zero.

Thus, the *SFIR* layer processes \hat{Z}_2 by two independent fully connected (FC) layers of 1024 units each. The FC layers are followed by a *hyperbolic tangent* and *sigmoid* function, whose outputs are the coefficient values and their index position respectively. To obtain the specific index position, the output of the *sigmoid* function is multiplied by T_s and a rounding down to the nearest integer is applied. This operation is not differentiable so we use an identity gradient as a backward pass approximation [29]. In order to have a high-quality reverberation, we use 2000 coefficients per second [26], thus, $T_s = 8$ samples for a sampling rate of 16 kHz.

The **synthesis back-end** uses the *SFIR* output \hat{Z}_3 , the envelopes \hat{Z}_1 and the residual connection R to synthesize the waveform and accomplish the reverberation task. It consists of an unpooling layer, a convolution and multiplication operation, a DNN with SAAFs (*DNN-SAAF*), two Squeeze-and-Excitation [30] LSTM layers (*SE-LSTM*) and a final convolutional layer.

Following the filter bank architecture: \hat{X}_4 is obtained by upsampling \hat{Z}_1 and the feature map \hat{X}_5 is accomplished by the locally connected convolution between the frequency band decomposition R and \hat{Z}_3 . The result of this convolution can be seen as explicitly modeling a frequency dependent reverberation response with the incoming audio. Furthermore, due to the temporal dependencies learnt by the Bi-LSTMs, \hat{X}_5 is able to represent from the onset response the late reflections of the reverberation task. Then the feature map \hat{X}_3 is the result of the element-wise multiplication of the reverberant response \hat{X}_5 and the learnt envelopes \hat{X}_4 . The envelopes are applied in order to avoid audible artifacts between input frames [27].

Secondly, the feature map \hat{X}_2 is obtained when the dynamic nonlinearities from the *DNN-SAAF* block are applied to R . The result of this operation consists of a learnt nonlinear transformation or waveshaping of the direct sound [19]. The *DNN-SAAF* block consists of 4 FC layers of 32, 16, 16 and 32 hidden units respectively. Each FC layer uses the *hyperbolic tangent* as nonlinearity except for the last one, which uses a SAAF layer.

Furthermore, we propose an *SE-LSTM* block to act as a time-varying gain for \hat{X}_2 and \hat{X}_3 . Since Squeeze-and-Excitation (SE) blocks explicitly and adaptively scale the channel-wise information of feature maps [30], we incorporate an LSTM layer in the SE architecture in order to include long-term context from the input. Each *SE-LSTM* builds on the architecture from [31], it consists of an *absolute value* operation and global average pooling operation followed by one LSTM and two FC layers of 32, 512 and 32 hidden units respectively. The LSTM and first FC layer are followed by a rectifier linear unit, while the last FC layer uses a *sigmoid* activation function. The *absolute value* is incorporated before the global average pooling since the feature maps are based on time-domain waveforms. Each *SE-LSTM* block process each feature map \hat{X}_2 and \hat{X}_3 , thus, applying a frequency dependent time-varying mixing gain where outputs are added together in order to obtain \hat{X}_0 .

The last layer corresponds to the *deconvolution* operation which

¹<https://mchijmma.github.io/modeling-plate-spring-reverb/>

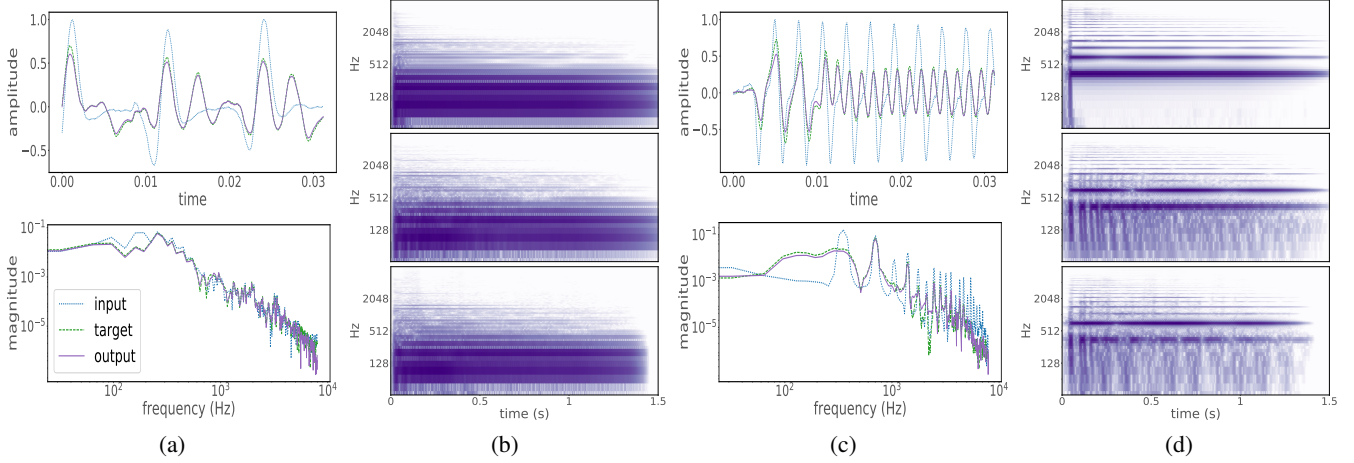


Fig. 2: Results for *model-1* with the test dataset. 2a and 2c) show a segment of the input, target and output frames and their respective FFT magnitudes for *plate* and *spring* reverb, respectively. 2b) *plate* and 2d) *spring* reverb and from top to bottom: input, target and output spectrograms of the test samples; color intensity represents higher magnitude.

is not trainable since its filters are the transposed weights of *Conv1D*. The complete waveform is synthesized using a *hann* window and constant overlap-add gain. All convolutions are along the time dimension and all strides are of unit value. Overall, each SAAF is locally connected and each function consists of 25 intervals between -1 to 1 and each Bi-LSTM and LSTM have dropout and recurrent dropout rates of 0.1 .

2.2. Training

Prior to training the whole model, as an initialization step, only the weights of *Conv1D* and *Conv1D-Local* are trained. Thus, within an unsupervised learning task, the adaptive front-end is able to process and reconstruct both the dry audio x and target audio y . This pretraining allows to have a better fitting when training for the reverb task. During this step the unpooling layer of the back-end uses the time positions of the maximum values recorded by the *max-pooling* operation. Once the front-end is initialized, all the weights of the convolutional, recurrent, dense and activation layers are trained following an end-to-end supervised learning task. The loss function to be minimized is based in time and frequency and described by:

$$loss = \alpha_1 \cdot mae(y, \hat{y}) + \alpha_2 \cdot mse(Y, \hat{Y}) \quad (1)$$

Where *mae* is the mean absolute error and *mse* is the mean squared error. Y and \hat{Y} are the log power magnitude spectra of the target and output respectively, and y and \hat{y} their respective waveforms. Prior to calculating the *mae*, a pre-emphasis filter $H(z) = 1 - 0.95z^{-1}$ is applied to y and \hat{y} , in order to add more weight to high frequencies [20]. We use a 4096-point Fourier transform (FFT) to obtain Y and \hat{Y} . In order to scale the time and frequency losses, we use 1.0 and $1e-4$ as the loss weights α_1 and α_2 respectively. Explicit minimization in the frequency and time domains resulted crucial when modeling such complex responses.

For both training steps, *Adam* [32] is used as optimizer and we use an early stopping patience of 25 epochs if there is no improvement in the validation loss. Afterwards, the model is fine-tuned further with the learning rate reduced by 25% and also a patience of 25 epochs. The initial learning rate is $1e-4$ and the batch size consists of the total number of frames per audio sample. We select the model with the lowest error for the validation subset.

2.3. Dataset

Plate reverbation is obtained from the *IDMT-SMT-Audio-Effects* dataset [33], which corresponds to individual 2-second notes and covers the common pitch range of various electric guitars and bass guitars. We use raw and *plate* reverb notes from the bass guitar recordings. *Spring* reverbation samples are obtained by processing the electric guitar raw audio samples with the *spring* reverb tank *Accutronics 4EB2C1B*.

For each reverb task we use 624 raw and effected notes and both the test and validation samples correspond to 5% of this subset each. The recordings are downsampled to 16 kHz and amplitude normalization is applied. Also, since the *plate* reverb samples have a fade-out applied in the last 0.5 seconds of the recordings, we process the *spring* reverb samples accordingly.

3. RESULTS & ANALYSIS

In order to compare the performance of the proposed architecture (*model-1*), we use the network from [23] (*model-2*), which has proven capable of modeling electromechanical devices such as the Leslie speaker. The latter presents an architecture similar to *model-1*, although its latent-space and back-end have been designed to explicitly learn and apply a modulation in order to match modulation based audio effects [34]. Both models are trained under the same procedure, tested with samples from the test dataset and the audio results are available online¹. Table 1 shows the corresponding *loss* values. The number of parameters for *model-1* and *model-2* are 410,977 and 275,073 and the time each model takes to process a 2 second audio sample is 0.752 and 0.4066 seconds, respectively. This

Table 1: *loss* values for *plate* and *spring* reverb models when tested with the test dataset.

Reverb	model	<i>mae</i>	<i>mse</i>	<i>loss</i>
plate	1	0.00214	7.75815	0.00292
	2	0.00316	27.08704	0.00587
spring	1	0.00366	9.43629	0.00461
	2	0.00474	33.09621	0.00805

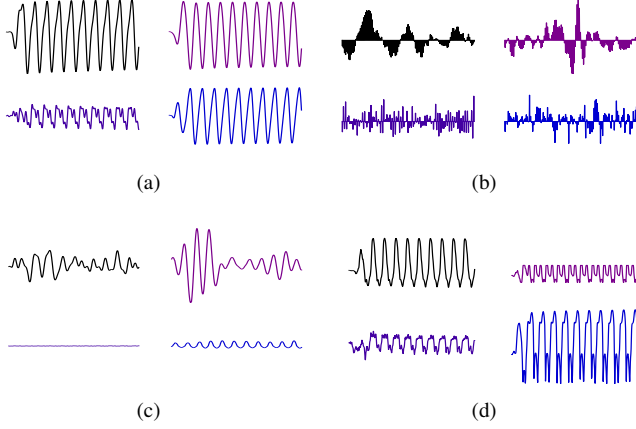


Fig. 3: Various internal plots for *model-1* from the test dataset of the *spring* modeling task. 3a) 4 rows from the frequency band decomposition \mathbf{R} . 3b) From $\hat{\mathbf{Z}}_3$, corresponding 4 sparse FIR filters learned by the latent-space. Following the filter bank architecture, 3c) and 3d) show the corresponding 4 rows from $\hat{\mathbf{X}}_{1,1}$ and $\hat{\mathbf{X}}_{1,2}$ respectively. Vertical axes are unitless and horizontal axes are time.

using a *Titan XP GPU* and non real-time *python* implementation.

The proposed model outperforms *model-2* in both tasks. For both reverb tasks and from the test subset, Fig. 2 shows selected input, target, and output waveforms together with their respective spectrograms. It can be seen that *model-1* matches very closely the target in the time and frequency domains. From the spectrograms, the smooth noise-like response of the *plate* and the dispersive reflections of the *spring* are noticeable. Overall, the initial onset responses are being modeled more accurately, whereas the late reflections differ more prominently in the case of the *spring*, which across models presents a higher loss. These differences in the frequency domain also correspond to the larger differences between the *mse* and the *mae*, thus, further exploration of the loss weights can be conducted.

Fig. 3 depicts internal plots of *model-1* when processing the frames from Fig. 2c. It shows how the model processes the input frame into the frequency band decomposition \mathbf{R} and learns a set of sparse FIR filters $\hat{\mathbf{Z}}_3$ for each frequency band. Then, the frequency dependent reverberation response $\hat{\mathbf{X}}_{1,1}$ is obtained by applying the filters and envelopes to \mathbf{R} . The nonlinear transformation of the direct sound $\hat{\mathbf{X}}_{1,2}$ is accomplished through the learnt waveshapers. These two representations are added together via a time-varying mixing gain, which is fed to the last layer so the audio waveform is reconstructed in the same manner as the front-end that decomposed it.

3.1. Listening test

Thirty participants between the ages of 23 and 46 took part in the experiment which was conducted at a professional listening room at Queen Mary University of London. The subjects were among musicians, sound engineers or experienced in critical listening. The audio was played via *Beyerdynamic DT-770 PRO* studio headphones and the Web Audio Evaluation Tool [35] was used to set up the test.

The participants were presented with samples from the test subset. Each page contained a reference sound, i.e. from the original *plate* or *spring* reverb. Participants were asked to rate 4 different samples according to the similarity of these in relation to the reference sound. The aim of the test was to identify which sound is closer to the reference. The samples consisted of outputs from *model-1*, *model-2*, a hidden copy of the reference and a dry sample as hidden

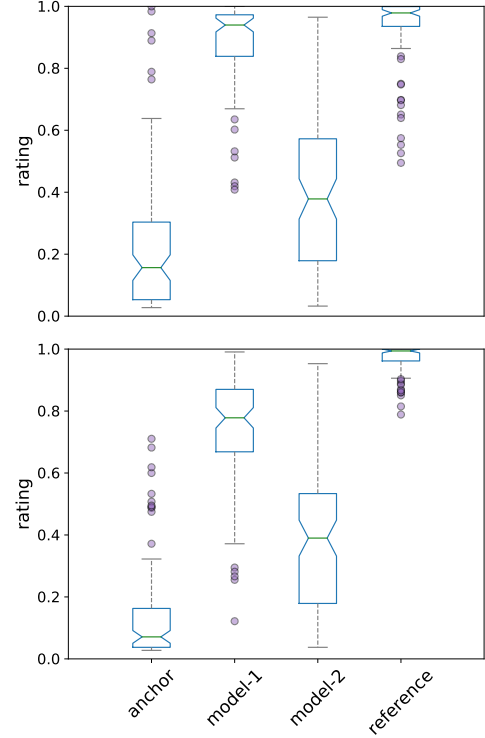


Fig. 4: Box plot showing the rating results of the listening tests. From top to bottom: *plate* and *spring* reverb tasks.

anchor. Thus, the test is based on the MUSHRA method [36].

The results of the listening test can be seen in Fig. 4 as a notched box plot. The end of the boxes represents the first and third quartiles, the end of the notches represents a 95% confidence interval, the green line depicts the median rating and the circles represent outliers. As expected, both anchor and reference have the lowest and highest median respectively. It can be seen that for both *plate* and *spring* reverb tasks, *model-1* is rated highly whereas *model-2* fails to accomplish the reverberation task. Thus, the perceptual findings confirm the results obtained with the *loss* metric and likewise, *plate* models have a better matching than *spring* reverberators. The rating and *loss* values for *spring* do not represent a significant decrease of performance, nevertheless, the modeling of *spring* late reflections could be further explored via a larger number of filters, different loss weights or input frame sizes.

4. CONCLUSION

In this work, we introduced a signal processing-informed deep learning architecture for modeling artificial reverberators. We explored the capabilities of learning sparse FIR filters and time-varying mixing gains within a DNN framework. We show the model successfully matching nonlinear time-varying transformations such as *plate* and *spring* reverb. Listening test results show that the model emulates closely the electromechanical devices and outperforms other DNNs for black-box modeling of audio effects. As future work, a parametric model, longer decay times and late reflections, and real-time implementations could be explored together with applications beyond effects modeling such as automatic reverberation and mixing.

5. REFERENCES

- [1] Udo Zölzer, *DAFX: digital audio effects*, John Wiley & Sons, 2011.
- [2] Vesa Välimäki et al., “Fifty years of artificial reverberation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.
- [3] Walter Kuhl, “The acoustical and technological properties of the reverberation plate,” *E. B. U. Review*, vol. 49, 1958.
- [4] Stefan Bilbao, *Numerical sound synthesis*, Wiley Online Library, 2009.
- [5] Julian Parker and Stefan Bilbao, “Spring reverberation: A physical perspective,” in *12th International Conference on Digital Audio Effects (DAFx-09)*, 2009.
- [6] Stefan Bilbao, Kevin Arcas, and Antoine Chaigne, “A physical model for plate reverberation,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006.
- [7] Stefan Bilbao, “A digital plate reverberation algorithm,” 2007, vol. 55, pp. 135–144.
- [8] Kevin Arcas et al., “On the quality of plate reverberation,” *Applied Acoustics*, vol. 71, no. 2, pp. 147–156, 2010.
- [9] Silvin Willemsen, Stefania Serafin, and Jesper R Jensen, “Virtual analog simulation and extensions of plate reverberation,” in *14th Sound and Music Computing Conference*, 2017.
- [10] Jonathan S Abel, David P Berners, and Aaron Greenblatt, “An emulation of the emt 140 plate reverberator using a hybrid reverberator structure,” in *127th Audio Engineering Society Convention*, 2009.
- [11] Aaron B Greenblatt, Jonathan S Abel, and David P Berners, “A hybrid reverberation crossfading technique,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010.
- [12] Keun Sup Lee, Nicholas J Bryan, and Jonathan S Abel, “Approximating measured reverberation using a hybrid fixed/switched convolution structure,” in *13th International Conference on Digital Audio Effects (DAFx-10)*, 2010.
- [13] Jonathan S Abel et al., “Spring reverb emulation using dispersive allpass filters in a waveguide structure,” in *121st Audio Engineering Society Convention*, 2006.
- [14] Stefan Bilbao and Julian Parker, “A virtual model of spring reverberation,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 4, pp. 799–808, 2009.
- [15] Stefan Bilbao, “Numerical simulation of spring reverberation,” in *16th International Conference on Digital Audio Effects (DAFx-13)*, 2013.
- [16] Vesa Välimäki, Julian Parker, and Jonathan S Abel, “Parametric spring reverberation effect,” *Journal of the Audio Engineering Society*, vol. 58, no. 7/8, pp. 547–562, 2010.
- [17] Julian Parker, “Efficient dispersion generation structures for spring reverb emulation,” *EURASIP Journal on Advances in Signal Processing*, 2011.
- [18] Marco A Martínez Ramírez and Joshua D Reiss, “End-to-end equalization with convolutional neural networks,” in *21st International Conference on Digital Audio Effects (DAFx-18)*, 2018.
- [19] Marco A Martínez Ramírez and Joshua D Reiss, “Modeling of nonlinear audio effects with end-to-end deep neural networks,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2019.
- [20] Eero-Pekka Damskägg et al., “Deep learning for tube amplifier emulation,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2019.
- [21] Alec Wright et al., “Real-time black-box modelling with recurrent neural networks,” in *22nd International Conference on Digital Audio Effects (DAFx-19)*, 2019.
- [22] Scott H Hawley et al., “Signaltrain: Profiling audio compressors with deep neural networks,” in *147th Audio Engineering Society Convention*, 2019.
- [23] Marco A Martínez Ramírez, Emmanouil Benetos, and Joshua D Reiss, “A general-purpose deep learning approach to model time-varying audio effects,” in *22nd International Conference on Digital Audio Effects (DAFx-19)*, 2019.
- [24] Xue Feng, Yaodong Zhang, and James Glass, “Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2014.
- [25] Kun Han et al., “Learning spectral mapping for speech dereverberation and denoising,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 23, no. 6, pp. 982–992, 2015.
- [26] Per Rubak and Lars G Johansen, “Artificial reverberation based on a pseudo-random impulse response II,” in *106th Audio Engineering Society Convention*, 1999.
- [27] Hanna Järveläinen and Matti Karjalainen, “Reverberation modeling using velvet noise,” in *30th Audio Engineering Society International Conference*, 2007.
- [28] Le Hou et al., “Convnets with smooth adaptive activation functions for regression,” in *Artificial Intelligence and Statistics*, 2017.
- [29] Anish Athalye, Nicholas Carlini, and David Wagner, “Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples,” in *International Conference on Machine Learning*, 2018.
- [30] Jie Hu, Li Shen, and Gang Sun, “Squeeze-and-excitation networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [31] Taejun Kim, Jongpil Lee, and Juhan Nam, “Sample-level cnn architectures for music auto-tagging using raw waveforms,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2018.
- [32] Diederik Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations. ICLR*, 2015.
- [33] Michael Stein et al., “Automatic detection of audio effects in guitar and bass recordings,” in *128th Audio Engineering Society Convention*, 2010.
- [34] Joshua D Reiss and Andrew McPherson, *Audio effects: theory, implementation and application*, CRC Press, 2014.
- [35] Nicholas Jillings et al., “Web Audio Evaluation Tool: A browser-based listening test environment,” in *12th Sound and Music Computing Conference*, 2015.
- [36] International Telecommunication Union, “Recommendation ITU-R BS.1534-1: Method for the subjective assessment of intermediate quality level of coding systems,” 2003.